User's Manual for the Gumbix Algorithm Suite

Markus Gumbel

2008-2011

Abstract

This paper describes the features of the Gumbix Algorithm Suite (GAS). This suite contains tools for the visualiziation of sorting algorithms and for different types of search trees. A ready-to-go Eclipse project helps to easily add new sort algorithms. This should enable students of the first semesters to integrate their own sorting algorithms into the gumbix-algorithm tool. Finally, a grammar tool draws derivation trees of Backus-Naur or context free grammars.

Contents

1 Introduction **2**
- 1.1 Overview . . . 2
- 1.2 Installation . . . 2

2 Sorting **3**
- 2.1 Overview . . . 3
- 2.2 Creating data for sorting . . . 4
- 2.3 Options for visualization . . . 4
- 2.4 Sound . . . 5
- 2.5 Algorithms . . . 5

3 Eclipse package for sorting **5**
- 3.1 Integration of a new sort algorithm . . . 5
- 3.2 Callbacks to the sorting tool . . . 5

4 Trees **7**

5 Grammars **7**
- 5.1 Overview . . . 7
- 5.2 Grammars . . . 7
- 5.3 Visualization of the derivation tree . . . 8

A Ressources **9**
- A.1 Download . . . 9
- A.2 Bug report . . . 9
- A.3 Contact . . . 9

1 Introduction

1.1 Overview

The Gumbix algorithm suite (GAS) consists of three different packages:

1. gumbix-algorithms tool
2. sorting algorithms as ready-to-go Eclipse project
3. grammar tool

The 1) **gumbix-algorithms tool** contains two executables for a) the demonstration and visualiziation of sorting algorithms and b) for different types of search trees. 2) The sorting algorithms are applied in a separate **Eclipse package**. This should enable students of the first semesters to integrate their own sorting algorithms into the gumbix-algorithm tool. Finally, 3) the **grammar tool** draws derivation trees of Backus-Naur or context free grammars. [1]

1.2 Installation

Prerequisite In any case, you need a Java Runtime Environment for JDK 1.5 or higher installed on your computer. The `java` command must be in the `PATH`.

gumbix-algorithm tool Extract the zip file (e.g. `gumbix-algorithms-1.4.0.zip`) in a directory on your file system, e.g. under `c:\gumbix-algorithms`. This directory will contain the following files and directories:

- `doc`: This user manual and other documentation
- `lib`: required libraries
- `sortUI.bat`: Start this batch file to run the sorting demo
- `tree.bat`: Start this batch file to run the search tree demo
- `algorithms.groovy`: Add your own sorting algorithms here. This approach is deprecated – so forget about it. See section 1 for a better way.

Eclipse project Extract the zip file (e.g. `gumbix-algorithms-sortext-1.4.0.zip`) in a directory on your file system, e.g. under `c:\gumbix-sortext`. This directory will contain the following files and directories:

- `doc`: This user manual and other documentation
- `lib`: required libraries
- `src`: contains a java file `MeinBubbleSort.java`
- `.classpath` and `.project`: Eclipse project definition

Now import this project into Eclipse (Typically via *File/Import*, then *Existing Projects into Workspace*). Now read section 3 how to write your own sorting algorithms.

[1] The grammar tool is not part of the gumbix-algorithm tool due to technical reasons only. gumbix-algorithm tool is written in Groovy whereas the newer grammar tool is written in Scala.

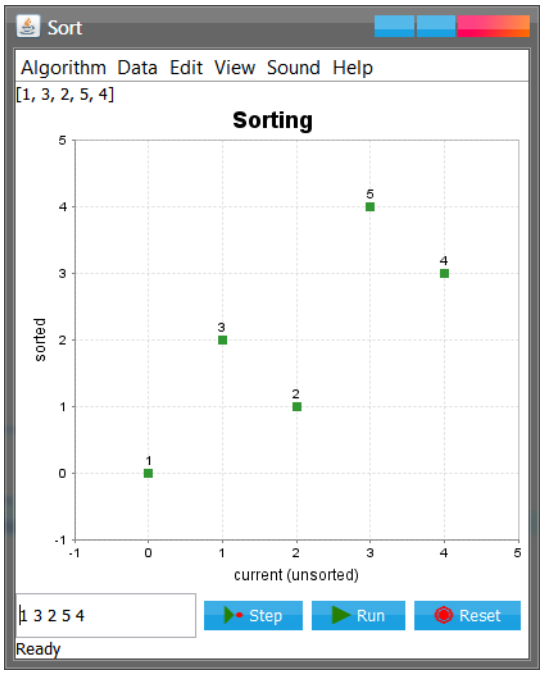

Figure 1: Sorting tool: Overview

grammar tool: Extract the zip file (e.g. `grammartree-1.1.0.zip`) in a directory on your file system, e.g. under `c:\grammartree`. This directory will contain the following files and directories:

- `doc`: This user manual and other documentation
- `examples`: Example grammars which can be loaded into the grammar tool
- `lib`: required libraries
- `grammartree.bat`: Start this batch file to run the grammar tool

2 Sorting

Note: If you want to integrate you own sorting algorithm, please also read section 3.

2.1 Overview

Start the demo by running `./sortUI.bat` (e.g. by double-clicking). This will open a similar window like shown in figure 1.

Visualization The panel shows a list of numbers which are ready to be sorted. The numbers are shown in the first row in their current order. Below you see a dot plot where the x-axis shows the current index-position of an element of the (probably unsorted) list. The y-axis shows the index-position of the same elements if they were already sorted. For instance, element 3 in figure 1 is currently at position 1 (see x-axis) and should be at position 2 (see y-axis) if the list was sorted. Note that the index positions

start at 0. So, a sorted list would result in a dot plot where all elements are on a diagonal line from (0, 0) to (n, n).

Running A list can be sorted by running the sort algorithm. The *Run*-button starts the execution and stops if the list is sorted, i.e. the algorithm has been completed. You may use the *Step*-button to step through the sorting process. You can switch between the run-mode and the step-mode by pressing the *Run*- and *Step*-buttons, respectively. The *Reset*-Button stops the execution.

2.2 Creating data for sorting

The sorting tool provides several ways for creating lists to sort. The menu *Data* shows the options. If a list of elements is automatically generated you may specify the size in the menu *Data/Size* beforehand.

1. *Text field*: Enter elements manually. The input field on the lower left corner lets you enter numbers. Each element must be separated by a space.
2. *Random*: Creates a list of random numbers. Note: the size of the list is specified in the menu *Data/Size* as mentioned above.
3. *Sorted*: Creates a list of already sorted numbers. The sorting order is ascending.
4. *Reverse*: Creates a list of descending sorted numbers.

If elements are entered manually (mode Text field) the sorting tool can interpret the values in two ways:

1. as numbers
2. as strings as they are

Select *Edit/Parse values as numbers* to activate option 1. Also, if you have created an interesting list, e.g. by creating a random list, you may copy the list into the text field where the numbers can be modified further. This can be accomplished with the menu *Edit/Copy current data to edit field*.

2.3 Options for visualization

The menu *View* contains further options for the visualization.

Delay Sorted lists are updated permanently when in run-mode. However, this happens typically too fast to follow. *View/Delay* sets the time interval (in ms) for a delay.

Inner and outer loops Typically, all sorting algorithms consist of two nested loops – sometimes hidden in recursion. These loops are denoted as inner and outer loop. However, looking at the changes while the algorithm processes the inner loop can sometimes be tedious and boring. You can tell the sorting tool not to display changes while performing the inner loop by deselecting *View/Show inner and outer loops*. Technical details on this topics can be found in section 3.2.

Show Index By default, the sorting tool displays the index position of the elements. However, if the lists consists of numbers only, it might also display the number's value directly. This option is configured in *View/Show Index*. Note that if non number values entered in the text field, the display mode must be *Show Index*.

2.4 Sound

The sorting process can not only be visualized, it can also be – well "audiolized". If you want to "hear" the list, select *Sound/Play Sound*. Now a number is interpreted as a MIDI[2]-note, i.e. the values must be in range 0 through 127. Note 60 is a Middle-C. Note that the range 30 to 90 sounds fine for humans. Thus, if sound is activated, the generation of example data is slightly different. Now the values are centered around the value 60, i.e. the note Middle-C. **Caution!** Playing very long lists (say more than 64 elements) will take some time. Currently, there is no way to interrupt the playing except of quitting the program.

2.5 Algorithms

Finally, the menu *Algorithm* lets you select the algorithm to use. The gumbix-algorithms tool comes with bunch of already implemented algorithms.

3 Eclipse package for sorting

This section tells you how to add your own sorting algorithm and run it in the sorting tool. The sorting tool itself is described in section 2.

3.1 Integration of a new sort algorithm

A simple way is to copy or modify the example `MeinBubbleSort.java` (found in `src`). Create a new class (here `MeinBubbleSort`) that has a main method to run. Also, create a class that inherits from `AbstractIntArraySort` (this is in our example `MeinBubbleSort`, too). Snippet 1 shows the main method and the way how a new algorithm is added. An object of type `Main` is created and the constructor gets the new algorithm and a name passed as parameters. This will open the sorting tool and add the algorithm in the *Algorithms*-menu.

Extending the `AbstractIntArraySort` requires to implement a `sort()`-method. Here the array to be sorted is passed in from the sorting tool as an argument. The initial array is created as described in section 2.2. Snippet 2 shows an example for the sort method.

Now you can implement your own sorting algorithm within this method.

3.2 Callbacks to the sorting tool

The sorting tool needs to know when to refresh the dot plot panel. Invoke the method `currentList()` anytime when your algorithm has performed a relevant sorting

[2]musical instrument digital interface

Algorithm 1 Adding a new algorithm and starting the sorting tool

```
public static void main(String[] args) {
  // 1. Parameter ist Ihre neue eigene Klasse.
  // 2. Parameter ist eine kurze Beschreibung, die
  // so später im Menu angezeigt wird
  Main main =
    new Main(new MeinBubbleSort(), "BubbleSort (von mir)");
}
```

Algorithm 2 `sort()`-method

```
protected void sort(int[] array) {
  // --- Ab hier können Sie Ihren eigenen Source-Code eingeben: ---
  for (int m = 0; m < array.length - 1; m++) {
    for (int i = 0; i < array.length - 1; i++) {
      if (array[i] > array[i + 1]) {
        int h = array[i];
        array[i] = array[i + 1];
        array[i + 1] = h;
      }
      // Mit dieser Anweisung zählen Sie eine Anweisung
      // in einer inneren Schleife um 1 hoch:
      currentList(array, INNER);
    }
    // Mit dieser Anweisung zählen Sie eine Anweisung
    // in einer äußeren Schleife um 1 hoch:
    currentList(array, OUTER);
  }
  // --- Bis hier können Sie Ihren eigenen Source-Code eingeben. ---
}
```

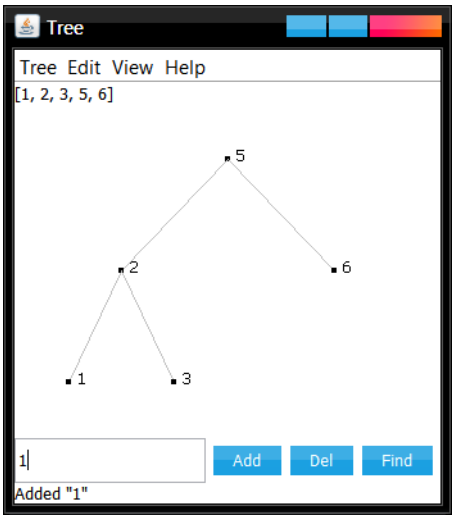

Figure 2: Overview of the tree tool

step. The bubble sort in the example above has an inner and outer loop. Thus, we callback the sorting tool accordingly. **Caution!** If your algorithm never calls the `currentList()`method, the sorting tool will also never update the sorting process before the sorting is done.

4 Trees

Start the demo by running `./treeUI.bat` (e.g. by double-clicking). This will open a similar window like shown in figure 2.

More documentation will follow soon!

5 Grammars

5.1 Overview

This tool reads context free grammars and generates all derivation trees up to a given tree depth. Note that this tool is still in beta status. Especially the error messages are not very helpful.

Start the demo by running `./grammartree.bat` (e.g. by double-clicking). This will open a similar window like in figure 3.

5.2 Grammars

A grammar can be loaded using the *File/Open* command. The tool understands two grammar formats:

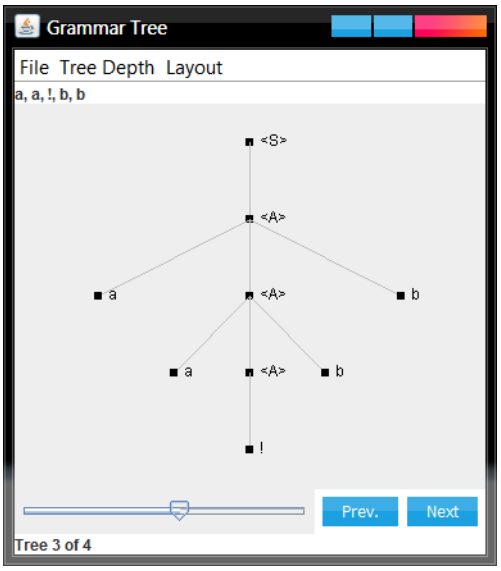

Figure 3: Grammar tool overview

1. Backus-Naur-Form (BNF) where a file ends with `.bnf` and

2. a standard grammar format where a file ends with `.grm`.

The directory `examples` contains a grammar in both styles which generates the words $a^n b^n$ ($n \geq 0$). Listings 1 and 2 show the grammar. This grammar is loaded by default.

Listing 1: Example in BNF format

```
<S> ::= <A>.
<A> ::= a <A> b.
<A> ::= !.
```

Listing 2: Example in standard format

```
S -> A.
A -> 'a' A 'b'.
A -> !.
```

The start symbol is always the rule S. The empty word is a !.

Constraints A grammar can have only one option per line. I.e. the | symbol to indicate alternatives is not yet implemented.

5.3 Visualization of the derivation tree

The derivation tree is shown in the main panel. The derived sentences is displayed on top of the tree. Each character is separated by a comma. The slider on the bottom enables you to browse through all derivation trees. Non-terminal symbols which are not yet expanded are cut off if the derivation depth exceeds the tree depth. They are then simply displayed as a non-terminal rule.

Tree Depth This number represents the depth of the tree which is generated and is specified in the menu *Tree Depth*. Note that many grammars yield a huge number of trees if the depth is greater than 4. Be careful – if the number of trees is to high the JVM will run out of memory.

A Ressources

A.1 Download

The distributables are available at `http://www.mi.hs-mannheim.de/gumbel/vorlesungen/ads/gumbix-algorithm-suite`.

A.2 Bug report

Any software has some bugs and especially this package is definitely not intended to be perfect. However, if you have found a bug or something that you find weird, please let me know and write to `m.gumbel@hs-mannheim.de`. An issue tracking system is not yet available.

A.3 Contact

Markus Gumbel, Mannheim University of Applied Sciences, Department of Computer Science, Institute for Medical Informatics. `m.gumbel@hs-mannheim.de` or `http://www.mi.hs-mannhe gumbel/`